# TES软件详细设计说明书

**所属项目： TES**

**版 本 号： V1.0**

**编 写 者： WJ**

**审 核 者： XC**

**批 注 者： XC**

**修订记录**

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 修订版本 | 修订描述 | 作者 |
| 2017-09 | V1.0 |  | 王建 |
|  |  |  |  |

目录

[**TES软件详细设计说明书** 1](#_Toc494275985)

[**1、** **引言** 6](#_Toc494275986)

[**1.1、** **编写目的** 6](#_Toc494275987)

[**1.2、** **背景** 6](#_Toc494275988)

[**1.3、** **参考资料** 6](#_Toc494275989)

[**1.4、** **术语定义说明** 7](#_Toc494275990)

[**1.5、** **运行环境概述** 7](#_Toc494275991)

[**2、** **总体方案确认** 7](#_Toc494275992)

[**3、** **部分系统UML类图** 8](#_Toc494275993)

[**4、** **Storm逻辑流程图** 9](#_Toc494275994)

[10](#_Toc494275995)

[**5、** **系统详细设计** 10](#_Toc494275996)

[**5.1、原始数据初始化** 11](#_Toc494275997)

[**5.1.1、描述** 11](#_Toc494275998)

[**5.1.2 功能实现** 11](#_Toc494275999)

[**5.1.3 功能程序入口** 11](#_Toc494276000)

[**5.1.4 代码** 12](#_Toc494276001)

[**5.2、维护工步的逻辑序号** 12](#_Toc494276002)

[**5.2.1、描述** 12](#_Toc494276003)

[**5.2.2、功能实现** 12](#_Toc494276004)

[**5.2.3、功能程序入口** 12](#_Toc494276005)

[**5.2.4、代码** 13](#_Toc494276006)

[**5.3、加载主数据** 13](#_Toc494276007)

[**5.3.1、描述** 13](#_Toc494276008)

[**5.3.2、功能实现** 13](#_Toc494276009)

[**5.3.3、功能程序入口** 14](#_Toc494276010)

[**5.3.4、代码** 14](#_Toc494276011)

[**5.4、 维护工步的逻辑循环号** 14](#_Toc494276012)

[**5.4.1、描述** 14](#_Toc494276013)

[**5.4.2、功能实现** 14](#_Toc494276014)

[**5.4.3、功能程序入口** 15](#_Toc494276015)

[**5.4.4、代码** 15](#_Toc494276016)

[**5.5、核心算法引擎** 15](#_Toc494276017)

[**5.5.1、描述** 15](#_Toc494276018)

[**5.5.2、 功能实现** 16](#_Toc494276019)

[**5.5.3、代码** 16](#_Toc494276020)

[**5.6、核心计算模块** 16](#_Toc494276021)

[**5.6.1、描述** 16](#_Toc494276022)

[**5.6.2、功能实现** 16](#_Toc494276023)

[**5.6.3、功能程序入口** 17](#_Toc494276024)

[**5.6.4、代码** 17](#_Toc494276025)

[**5.7、计算后处理模块** 17](#_Toc494276026)

[**5.7.1、描述** 17](#_Toc494276027)

[**5.7.2、功能实现** 18](#_Toc494276028)

[**5.8、业务序号维护模块** 18](#_Toc494276029)

[**5.8.1、描述** 18](#_Toc494276030)

[**5.8.2、功能实现** 18](#_Toc494276031)

[**5.9、数据同步模块** 18](#_Toc494276032)

[**5.9.1、描述** 18](#_Toc494276033)

[**5.9.2、功能实现** 18](#_Toc494276034)

[**5.9.3、功能程序入口** 19](#_Toc494276035)

[**5.9.4、代码** 19](#_Toc494276036)

[**5.10、Redis模块** 20](#_Toc494276037)

[**5.10.1、描述** 20](#_Toc494276038)

[**5.10.2、功能实现** 20](#_Toc494276039)

[**5.11、Hana模块** 20](#_Toc494276040)

[**5.11.1、描述** 20](#_Toc494276041)

[**5.11.2、功能实现** 20](#_Toc494276042)

[**5.12、二级缓存模块** 21](#_Toc494276043)

[**5.12.1、描述** 21](#_Toc494276044)

[**5.12.2、功能实现** 21](#_Toc494276045)

[**5.12.3、程序功能入口** 21](#_Toc494276046)

[**5.12.4、代码** 22](#_Toc494276047)

1. **引言**

随着业务的不断扩展和测试需求的不断增长，对测试结果的计算分析成为挑战，传统依靠人工处理的统计方法无法处理上千万级别的数据规模，实时性较低。科技不断在进步，利用大数据技术处理TB级数据显得至关重要，通过分布式流式计算框架、分布式高可用消息队列等大数据技术，可以很从容的解决目前用户的需求，同时降低人力的投入成本，对测试的效率以及分析计算的性能有非常大的提升。

* 1. **编写目的**

本需求的编写目的在于明确TES的开发途径以及应用方法，更合理、科学的使用大数据技术解决实际业务问题，以便于对系统进行使用和维护。

本需求的预期读者是TES相关的业务人员、开发人员以及系统运维人员；

* 1. **背景**

项目单位：时代新能源科技有限公司

主管部门：IT&TVC

* 1. **参考资料**

Apache Storm：<http://storm.apache.org/>

Apache Kafka：<http://kafka.apache.org/>

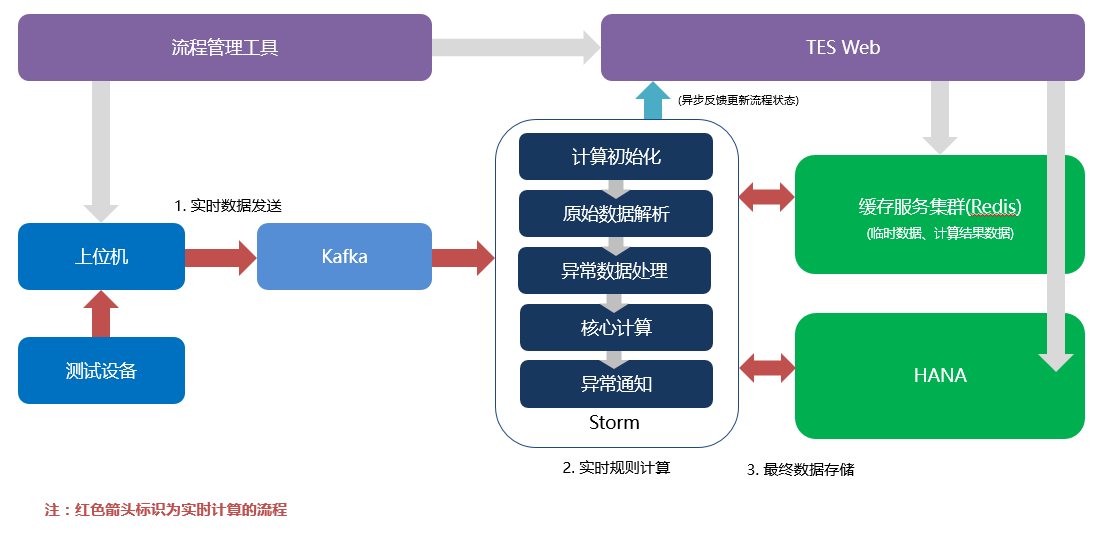
* 1. **术语定义说明**

|  |  |  |
| --- | --- | --- |
| 术语 | 定义说明 | 备注 |
| 算法 | 对测试场景的判异规则算法，算法内容是一段逻辑代码 |  |
| 测试数据 | 电池测试过程的数据，包含电池相关信息，测试过程数据以及相关指标结果 |  |
| 主数据 | 电池测试的主数据信息，包含测试流程信息、工步信息以及算法等内容； |  |
| 二级缓存 | 在本次方案中是基于Google cache框架实现，是相对于Redis（一级缓存）的延展模块 |  |
| POJO | Java bean 对象 |  |
| Spout | Storm数据流入口 |  |
| Bolt | Storm数据处理对象 |  |
| FastJson | 阿里的json处理器 |  |

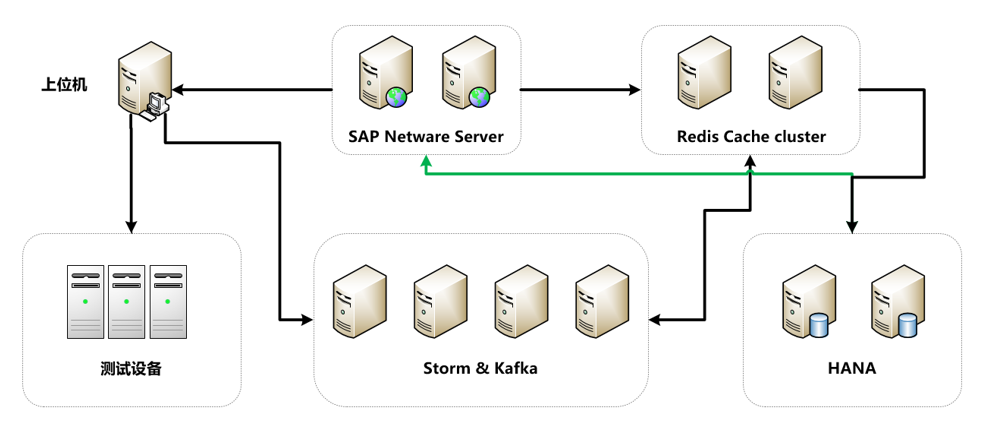
* 1. **运行环境概述**

|  |  |  |
| --- | --- | --- |
| 运行环境 | 版本要求 | 描述 |
| Java | 1.7 |  |
| Python | 2.6 |  |
| OS | Linux CentOS/Redhat：6.8 |  |

1. **总体方案确认**



应用架构图

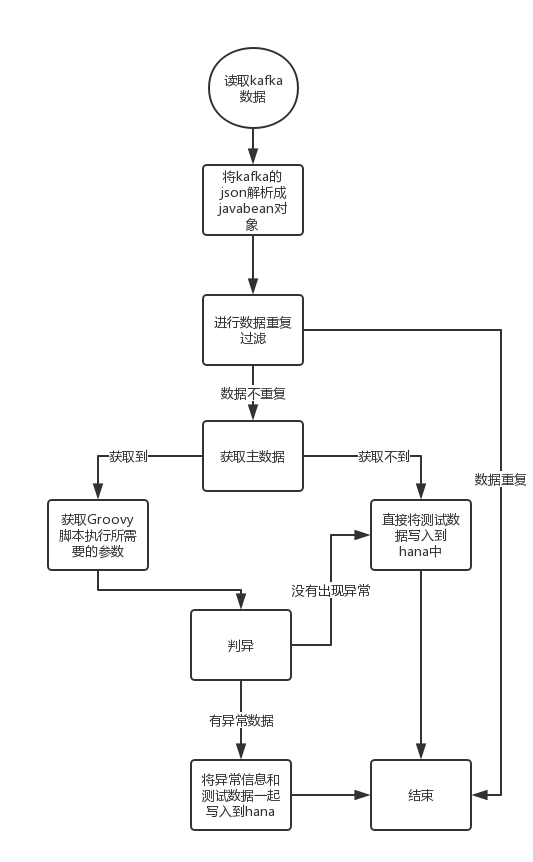


硬件架构图

1. **部分系统UML类图**



1. **Storm逻辑流程图**



1. **系统详细设计**

**5.1、原始数据初始化**

**5.1.1、描述**

将新威机发送过来的测试原始数据转换成POJO对象（便于后面程序中使用），并进行去重（去重的作用是防止数据源头有重复数据发送过来，做不必要的处理）。

**5.1.2 功能实现**

Storm通过spout对象去取新威机发送到kafka中的测试数据，并将测试数据转化成Tuple对象传递给bolt对象，bolt对象调用execute方法进行处理，通过使用google的fastJson框架将测试数据转化成POJO对象。

过滤工况数据：将需要过滤的工况数据的resouceID存放到redis中，程序第一次加载时会将工况数据的resouceID全部加载到缓存中，这样可减少系统和redis的交互次数，提高系统性能。当流过来的测试数据的resouceID和需要过滤的resouceID匹配上时，则将这条数据过滤掉。

数据去重：数据去重是根据数据的messageID来去重的，对每条数据设置一个messageID（remark+resouceID+channelID+sequenceID+数据的相对时间+数据的绝对时间）。将每条数据的messageID存放到redis缓存中，当下一条数据过来时首先查看redis中存不存在已有的数据，如果有则直接返回，不做判异。

### 5.1.3 功能程序入口

com.edcs.tds.storm.service. DataHandler. initRequestMessage()

**5.1.4 代码**



**5.2、维护工步的逻辑序号**

**5.2.1、描述**

维护工步的逻辑序号是为了后期会出现通过当前的数据去找该条数据的上一个工步的最后一条数据。

**5.2.2、功能实现**

对于同一个remark，当遇到数据的pvDataFlag=89（表明是某个工步的第一条数据）时，则工步的逻辑序号加1，每个流程的逻辑序号是维护在redis缓存中的。

### 5.2.3、功能程序入口

com.edcs.tds.storm.service. DataService. updateStepLogicNumber()

**5.2.4、代码**



**5.3、加载主数据**

**5.3.1、描述**

加载主数据是为了后期需要获取主数据中的信息来做判异使用。为了性能考虑，加载主数据时先从代码缓存中获取，如果获取不到再到hana数据库中获取，获取到之后再加入到代码二级缓存中。

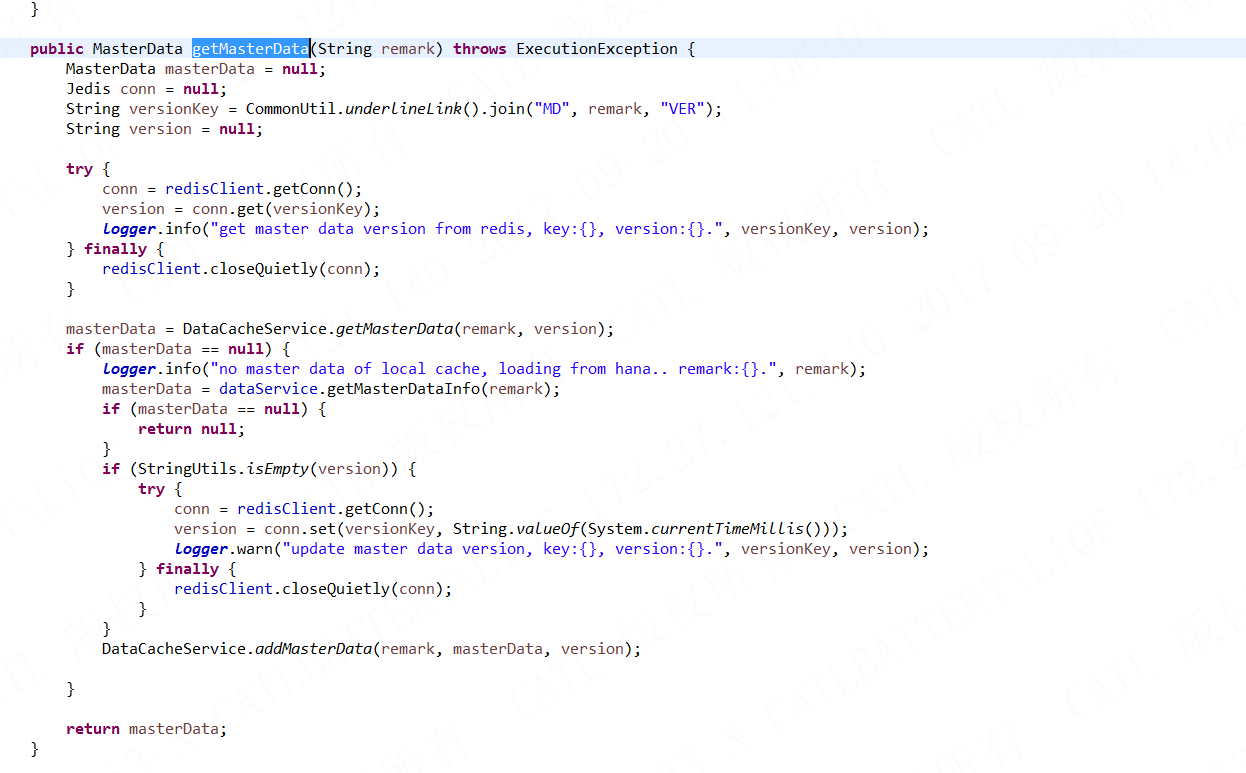
**5.3.2、功能实现**

加载主数据时，先到redis中获取该主数据对应的版本信息，通过版本信息到代码二级缓存中查找该版本的主数据有没有存放在缓存中（代码中的二级缓存是通过google的LoadingCache实现的，为了防止内存泄漏，缓存中每一个key都设置了不使用时的超时时长，超过这一时长没有被使用则被自动清理），如果没有在缓存中找到则需要到hana中重新读取最新的主数据信息，读取到后将该主数据的信息已经对应的版本信息全部缓存到二级缓存中，这样后续的同一remark的测试数据就不会到hana中读取主数据信息了。这不仅提高了系统的性能，而且降低了hana数据库的压力。

### 5.3.3、功能程序入口

com.edcs.tds.storm.topology.calc. CalcBolt. getMasterData()

**5.3.4、代码**



**5.4、 维护工步的逻辑循环号**

**5.4.1、描述**

由于上位机发送上来的数据的循环号不准确，所以需要storm程序自己去维护一套工步的逻辑循环号。供后续使用。维护是在redis中实现的。

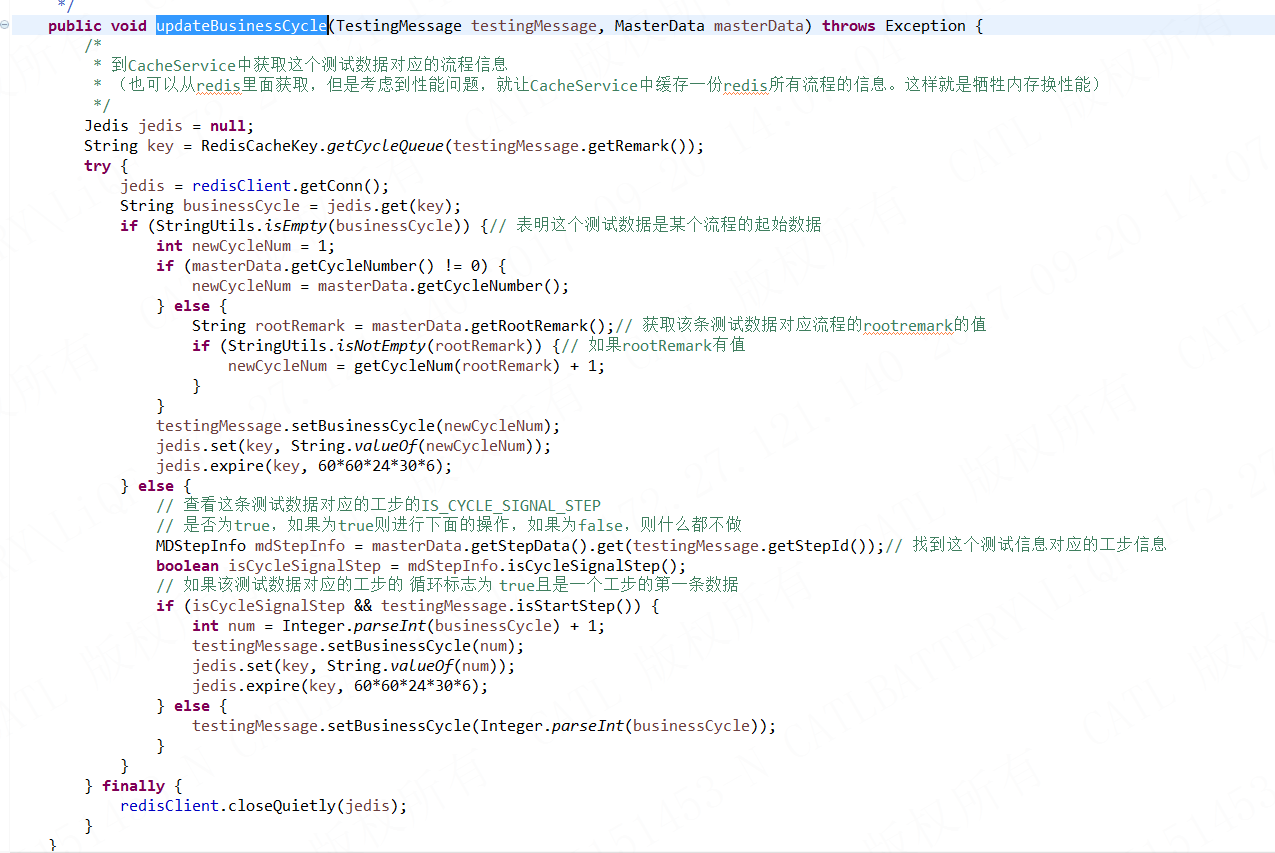
**5.4.2、功能实现**

当收到一条测试数据时，首先到redis中获取当前remark的循环信息，如果获取不到则说明是第一次维护，则首先通过remark来获取对应的主数据信息，看看主数据信息中有没有rootRemark值，如果没有则直接将该条数据的逻辑循环号设置成1，并写入到redis中，如果有值，则需要查询hana中所有remark等于rootRemark的逻辑循环号，并且取最大值，将这个最大值加1作为当前测试数据的逻辑循环号，并写入到redis中。如果从redis中获取到当前remark的逻辑循环号，则要根据当前测试数据的stepID到主数据中查询对应的stepID的工步信息，查看这个信息的isCycleSignalStep 是否为true，如果为true则当前的逻辑循环号需要加1，反之则不需要加1.

### 5.4.3、功能程序入口

com.edcs.tds.storm.service. DataService. updateBusinessCycle()

### 5.4.4、代码



**5.5、核心算法引擎**

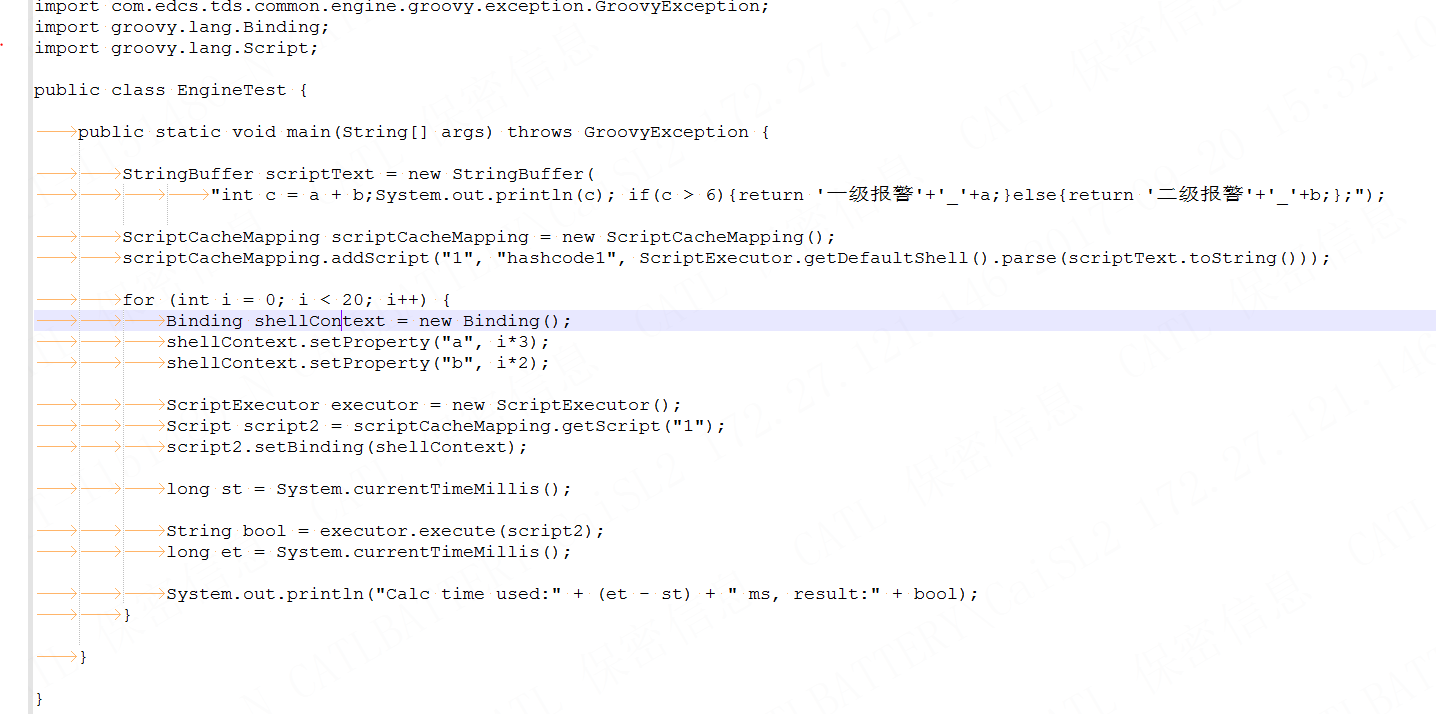
**5.5.1、描述**

核心算法引擎基于Groovy实现，Groovy是一种基于JVM（Java虚拟机）的敏捷开发语言，它结合了Python、Ruby和Smalltalk的许多强大的特性，Groovy 代码能够与 Java 代码很好地结合，也能用于扩展现有代码。由于其运行在 JVM 上的特性，Groovy 可以使用其他 Java 语言编写的库。

**5.5.2、 功能实现**

* 算法编译，支持对算法逻辑代码的编译，并且通过缓存机制对算法实例进行缓存，便于计算进行调用；
* 算法执行，支持对算法实例的调用执行，执行后返回计算结果；
* 异常处理，编译与执行过程异常记录日志；

### 5.5.3、代码



**5.6、核心计算模块**

**5.6.1、描述**

核心计算模块是整个核心计算的关键逻辑，通过解析相应的测试流程，加载对应的场景算法进行计算，计算结果如果包含异常值，则触发相应的邮件警告通知；

**5.6.2、功能实现**

* + 核心计算，加载本次测试的场景算法集合，调用算法执行，并且解析算法计算结果；
  + 邮件通知，如果算法计算结果满足条件，则调用邮件通知，发送相应内容的邮件，邮件的收件人由系统预先设置，通过数据初始化模块进行加载；

### 5.6.3、功能程序入口

com.edcs.tds.storm.topology.calc. RuleCalc. testingRuleCalc()

### 5.6.4、代码



**5.7、计算后处理模块**

**5.7.1、描述**

计算后处理是计算结束后的数据处理模块，包含对结果数据的封装（数据字段补充、报警单数据封装等）、二级缓存中过期数据的清理。计算完成后，结果数据和过程数据将通过数据同步模块存入数据库；

**5.7.2、功能实现**

* + 数据处理，当一个流程结束后，将清理二级缓存中不需要的主数据信息，通过生产报警单数据、封装结果数据等操作，这些数据将加入数据同步模块的操作队列中；

**5.8、业务序号维护模块**

**5.8.1、描述**

业务序号维护模块包含对逻辑序号、工步序号的维护和更新，每一条测试数据进行计算时会通过Redis模块维护该测试流程的逻辑序号、工步序号，对异常或者错误的序号值进行修正；

**5.8.2、功能实现**

* + 逻辑序号，获取当前测试数据的逻辑序号值并且判断是否是流程开始环节，如果不是则维护序号（自增操作），同时判断当前逻辑序号和缓存中的逻辑序号是否一致，不一致则修正逻辑序号；
  + 工步序号，获取当前测试数据的工步编号，并且维护工步序号，同时判断当前工步序号和缓存中的工步序号是否一致，不一致则修正工步序号；

**5.9、数据同步模块**

**5.9.1、描述**

数据同步模块是对计算结果数据的数据存储模块，数据同步模块中维护了一个数据暂存队列，当队列达到预设值时，调用Hana模块将数据批量存入数据库，数据同步模块是一个异步线程；

**5.9.2、功能实现**

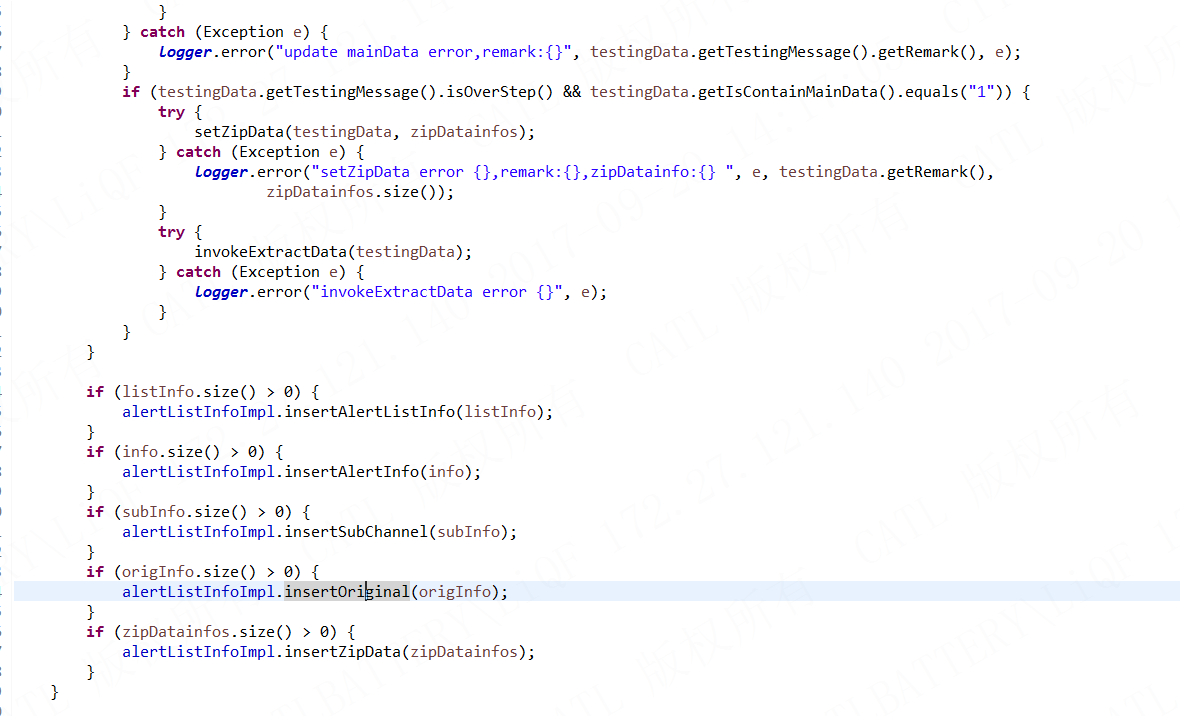
* + 数据暂存队列，维护一个集合，当计算结束后， 计算后处理模块会将结果数据存入队列，队列不设置内容上限；
  + 数据存储，判断是否达到队列预设值，如果达到预设值则调用Hana模块进行批量写操作；

### 5.9.3、功能程序入口

com.edcs.tds.storm.dao. ResultDataDao. setAllData()

### 5.9.4、代码





**5.10、Redis模块**

**5.10.1、描述**

Redis模块包含了对Redis的操作接口、Redis连接池，其他模块通过Redis连接池获取Redis Server连接，对Redis数据进行操作（增删改查）；

**5.10.2、功能实现**

* + 连接池，基于Java原生连接池框架实现一个Redis连接池，连接池包含对连接的维护、创建等管理功能，业务模块无需关心连接池内部机制，随取随用；

**5.11、Hana模块**

**5.11.1、描述**

Hana模块包含对Hana数据库的操作接口、JDBC连接池（基于DBCP实现），其他模块通过调用JDBC Helper模块（操作接口）进行数据库的更新与查询操作；

**5.11.2、功能实现**

* + 连接池，基于DBCP实现的通用连接池，连接池包含对连接的维护、创建等管理功能；
  + JDBC Helper，封装对Hana数据库的查询接口、更新接口，业务模块直接调用接口进行数据库操作；

**5.12、二级缓存模块**

**5.12.1、描述**

二级缓存模块（基于Google Cache实现）包含对应用程序内部缓存的存取、管理等功能，是Redis模块（一级缓存）的延展，在高并发计算的应用场景中，通过多级缓存机制可以有效降低IO的开销，提升系统整体的计算性能；

**5.12.2、功能实现**

* + 主数据缓存，维护和管理主数据信息，包含对数据元的生命周期管理，内存管理以及对外提供数据的存取接口。二级缓存模块可以容纳至少1.5万条主数据信息；

### 5.12.3、程序功能入口

com.edcs.tds.storm.cache

### 5.12.4、代码

